IN THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) In a computer system having a scalar processing unit and a vector processing unit, wherein the vector processing unit includes a vector dispatch unit, a method of decoupling operation of the scalar processing unit from that of the vector processing unit, the method comprising:

~~sending~~ dispatching a vector instruction from the scalar processing unit to the vector dispatch unit, wherein dispatching includes sending the vector instruction from the scalar processing unit to the vector dispatch unit even if all scalar operands are not ready; ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

reading [[a]] scalar ~~operand~~ operands from the scalar processing unit, wherein reading includes transferring the scalar ~~operand~~ operands from the scalar processing unit to the vector dispatch unit;

predispatching, within the vector dispatch unit, the vector instruction received from the scalar processing unit if all previously received vector instructions are scalar committed; ~~within the vector dispatch unit if the vector instruction is scalar committed;~~

dispatching the predispatched vector instruction from the vector dispatch unit if all required scalar operands are ready; and

executing the ~~dispatched~~ vector instruction dispatched from the vector dispatch unit as a function of the scalar ~~operand~~ operands.

2. (Previously Presented) The method according to claim 1, wherein executing the dispatched vector instruction includes translating an address associated with the vector instruction and trapping on a translation fault.

3. (Currently Amended) In a computer system having a scalar processing unit and a vector processing unit, wherein the vector processing unit includes a vector dispatch unit, a method of decoupling operation of the scalar processing unit from that of the vector processing unit, the method comprising:

~~sending~~ <u>dispatching</u> a vector instruction from the scalar processing unit to the vector dispatch unit, <u>wherein dispatching includes sending the vector instruction from the scalar processing unit to the vector dispatch unit even if all scalar operands are not ready;</u> ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

reading [[a]] scalar ~~operand~~ <u>operands from the scalar processing unit</u>, wherein reading includes transferring the scalar ~~operand~~ <u>operands</u> from the scalar processing unit to the vector dispatch unit;

predispatching<u>, within the vector dispatch unit,</u> the vector instruction <u>received from the scalar processing unit if all previously received vector instructions are scalar committed;</u> ~~within the vector dispatch unit if the vector instruction is scalar committed;~~

dispatching the predispatched vector instruction <u>from the vector dispatch unit</u> if all required <u>scalar</u> operands are ready;

generating an address for a vector load;

issuing a vector load request to memory;

receiving vector data from memory;

storing the vector data in a load buffer;

transferring the vector data from the load buffer to a vector register; and

executing the ~~dispatched~~ vector instruction <u>dispatched from the vector dispatch unit</u> on the vector data stored in the vector register.

4. (Original) The method according to claim 3, wherein the vector processing unit includes a vector execute unit and a vector load/store unit, wherein issuing a vector load request to memory includes issuing and executing vector memory references in the vector load/store unit

when the vector load store unit has received the instruction and memory operands from the scalar processing unit.

5. (Currently Amended) In a computer system having a scalar processing unit and a vector processing unit, wherein the vector processing unit includes a vector dispatch unit, a method of decoupling operation of the scalar processing unit from that of the vector processing unit, the method comprising:

~~sending~~ dispatching a vector instruction from the scalar processing unit to the vector dispatch unit, wherein dispatching includes sending the vector instruction from the scalar processing unit to the vector dispatch unit even if all scalar operands are not ready; ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

reading [[a]] scalar ~~operand~~ operands from the scalar processing unit, wherein reading includes transferring the scalar ~~operand~~ operands from the scalar processing unit to the vector dispatch unit;

predispatching, within the vector dispatch unit, the vector instruction received from the scalar processing unit if all previously received vector instructions are scalar committed; ~~within the vector dispatch unit if the vector instruction is scalar committed;~~

dispatching the predispatched vector instruction from the vector dispatch unit if all required scalar operands are ready;

generating a first and a second address for a vector load;

issuing first and second vector load requests to memory;

receiving vector data associated with the first and second addresses from memory;

storing vector data associated with the first address in a first vector register;

storing vector data associated with the second address in a second vector register;

executing a vector instruction on the vector data stored in the first vector register;

renaming the second vector register; and

executing the ~~dispatched~~ vector instruction dispatched from the vector dispatch unit on the vector data stored in the vector register.

6. (Previously Presented) The method according to claim 5, wherein the vector processing unit includes a vector execute unit and a vector load/store unit, wherein issuing a vector load request to memory includes issuing and executing vector memory references in the vector load/store unit when the vector load store unit has received the instruction and memory operands from the scalar processing unit.

7. (Currently Amended) A computer system, comprising:

a scalar processing unit; and

a vector processing unit, wherein the vector processing unit includes a vector dispatch unit, a vector execute unit and a vector load/store unit;

wherein the scalar processing unit ~~sends~~ dispatches a vector instruction ~~and a scalar operand~~ to the vector dispatch unit even if all scalar operands are not ready; ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

wherein the scalar processing unit reads scalar operands and transfers the read scalar operands from the scalar processing unit to the vector dispatch unit;

wherein the vector dispatch unit predispatches, within the vector dispatch unit, the vector instruction received from the scalar processing unit if all previously received vector instructions are scalar committed ~~within the vector dispatch unit if the vector instruction is scalar committed~~ and then dispatches the predispatched vector instruction from the vector dispatch unit to one or more of the vector execute unit and the vector load/store unit if all required scalar operands are ready;

wherein the vector load/store unit receives an instruction and memory operands from the scalar processing unit, issues and executes a vector memory load reference as a function of the instruction and the memory operands received from the scalar processing unit, and stores data received as a result of the vector memory reference in a load buffer; and

wherein the vector execute unit issues the vector memory load instruction and transfers the data received as a result of the vector memory reference from the load buffer to a vector register.

8. (Currently Amended) In a computer system having a scalar processing unit and a vector processing unit, a method of decoupling scalar and vector execution, comprising:

dispatching scalar instructions to a scalar instruction queue in the scalar processing unit;

~~sending~~ dispatching a vector instruction that requires scalar operands from the scalar processing unit to the scalar instruction queue and to a vector instruction queue, wherein dispatching includes sending the vector instruction from scalar processing unit to the vector dispatch unit even if all scalar operands are not ready; ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

executing the vector instruction in the scalar processing unit, wherein executing the vector instruction in the scalar processing unit includes writing a scalar operand to a scalar operand queue;

predispatching, within the vector dispatch unit, the vector instruction ~~sent~~ received from the scalar processing unit if all previously received vector instructions are scalar committed; ~~to the vector instruction queue if the vector instruction is scalar committed;~~

notifying the vector processing unit that the scalar operand is available in the scalar operand queue;

dispatching the predispatched vector instruction from the vector dispatch unit, if all required scalar operands are ready; and

executing the ~~dispatched~~ vector instruction dispatched from the vector dispatch unit in the vector processing unit, wherein executing the vector instruction in the vector processing unit includes reading the scalar operand from the scalar operand queue.

9. (Currently Amended) In a computer system having a scalar processing unit and a vector processing unit, a method of decoupling a vector memory reference and a vector execution, comprising:

dispatching scalar instructions to a scalar instruction queue in the scalar processing unit;

~~sending~~ dispatching a vector instruction from the scalar processing unit to the scalar instruction queue and to a vector instruction queue[[,]] in the vector processing unit, wherein dispatching includes sending the vector instruction from the scalar processing unit to the vector instruction queue in the vector processing unit even if all scalar operands are not ready; ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

executing the vector instruction in the scalar processing unit, wherein executing the vector instruction in the scalar processing unit includes generating an address and writing the address to a scalar operand queue;

predispatching, within the vector processing unit, the vector instruction ~~sent~~ received from the scalar processing unit if all previously received vector instructions are scalar committed; ~~to the vector instruction queue if the vector instruction is scalar committed;~~

notifying the vector processing unit that the address is available in the scalar operand queue;

dispatching the predispatched vector instruction from the vector processing unit if all required scalar operands are ready; and

executing the ~~dispatched~~ vector instruction dispatched from the vector processing unit in the vector processing unit, wherein executing the vector instruction in the vector processing unit includes reading the address from the scalar operand queue and generating a memory request as a function of the address read from the scalar operand queue.

10. (Currently Amended) In a computer system having a scalar processing unit and a vector processing unit, a method of executing a vector instruction, comprising:

dispatching scalar instructions to a scalar instruction queue in the scalar processing unit;

~~sending~~ dispatching a vector instruction from the scalar processing unit to the scalar instruction queue and to a vector instruction queue[[,]] in the vector processing unit, wherein dispatching includes sending the vector instruction from the scalar processing unit to the vector instruction queue in the vector processing unit even if all scalar operands are not ready; ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

executing the vector instruction in the scalar processing unit, wherein executing the vector instruction in the scalar processing unit includes generating an address and writing the address to a scalar operand queue;

predispatching, within the vector processing unit, the vector instruction ~~sent~~ received from the scalar processing unit if all previously received vector instructions are scalar committed; ~~to the vector instruction queue if the vector instruction is scalar committed;~~

notifying the vector processing unit that the address is available in the scalar operand queue;

dispatching the predispatched vector instruction from the vector processing unit if all required scalar operands are ready; and

executing the ~~dispatched~~ vector instruction dispatched from the vector processing unit in the vector processing unit, wherein executing the dispatched vector instruction in the vector processing unit includes:

reading the address from the scalar operand queue;

generating a memory request as a function of the address read from the scalar operand queue;

receiving vector data from memory;

storing the vector data in a load buffer;

transferring the vector data from the load buffer to a vector register; and

executing a vector instruction on the vector data stored in the vector register.

11. (Currently Amended) In a computer system having a scalar processing unit and a vector processing unit, a method of unrolling a loop, comprising:

preparing a first and a second vector instruction, wherein each vector instruction execute an iteration through the loop and wherein each vector instruction requires calculation of a scalar loop value;

~~sending~~ dispatching the first and second vector instructions from the scalar processing unit to ~~the~~ a scalar instruction queue in the scalar processing unit and to a vector instruction queue[[,]] in the vector processing unit, wherein dispatching includes sending the first and second vector instructions from the scalar processing unit to the vector processing unit even if all scalar operands are not ready; ~~wherein sending includes marking the vector instruction as complete if the vector instruction is not a vector memory instruction and if the vector instruction does not require scalar operands;~~

executing a portion of each vector instruction in the scalar processing unit, wherein executing a portion of each vector instruction in the scalar processing unit includes writing a scalar operand representing the scalar loop value calculated for each vector instruction to a scalar operand queue;

predispatching, within the vector processing unit, each vector instruction ~~sent~~ received from the scalar processing unit if all previously received vector instructions are scalar committed; ~~to the vector instruction queue if the vector instruction is scalar committed;~~

notifying the vector processing unit that the scalar operand is available in the scalar operand queue;

dispatching the predispatched vector instruction from the vector processing unit if all required scalar operands are ready; and

executing the first and second ~~dispatched~~ vector instructions dispatched from the vector processing unit in the vector processing unit, wherein executing the dispatched vector instruction in the vector processing unit includes reading the scalar operands associated with each instruction from the scalar operand queue.

12. (Previously Presented) The method according to claim 3, wherein storing the vector data in a load buffer and transferring the vector data from the load buffer to a vector register are decoupled from each other.

13. (Previously Presented) The method according to claim 3, wherein storing the vector data in a load buffer includes writing memory load data to the load buffer until all previous memory operations complete without fault.

14. (Previously Presented) The method according to claim 4, wherein storing the vector data in a load buffer and transferring the vector data from the load buffer to a vector register are decoupled from each other.

15. (Previously Presented) The method according to claim 4, wherein storing the vector data in a load buffer includes writing memory load data to the load buffer until all previous memory operations complete without fault.

16. (Previously Presented) The system according to claim 7, wherein the load buffer stores memory load data until it is determined that no previous memory operation will fail and, if no previous memory operations have failed, the load buffer transfers the data to the vector register.

17. (New) The method according to claim 1, wherein the method further includes: marking the vector instruction as complete, wherein marking the vector instruction as complete includes:

if the vector instruction is not a memory instruction and if the vector instruction does not require scalar operands, indicating that the vector instruction is complete when the vector instruction is dispatched from the scalar processing unit;

if the vector instruction is not a memory instruction but requires scalar operands, indicating that the vector instruction is complete when the scalar operands are available; and

if the vector instruction is a memory instruction, indicating that the vector instruction is complete when the vector address has been translated; and

graduating the vector instruction if the vector instruction is marked complete.

18. (New) The system according to claim 7, wherein the system further marks the vector instruction as complete, wherein marking the vector instruction as complete includes:

if the vector instruction is not a memory instruction and if the vector instruction does not require scalar operands, indicating that the vector instruction is complete when the vector instruction is dispatched from the scalar processing unit;

if the vector instruction is not a memory instruction but requires scalar operands, indicating that the vector instruction is complete when the scalar operands are available; and

if the vector instruction is a memory instruction, indicating that the vector instruction is complete when the vector address has been translated; and

wherein the system graduates the vector instruction if the vector instruction is marked complete.

19. (New) The method according to claim 8, wherein the method further includes: marking the vector instruction as complete, wherein marking the vector instruction as complete includes:

if the vector instruction is not a memory instruction and if the vector instruction does not require scalar operands, indicating that the vector instruction is complete when the vector instruction is dispatched from the scalar processing unit;

if the vector instruction is not a memory instruction but requires scalar operands, indicating that the vector instruction is complete when the scalar operands are available; and

if the vector instruction is a memory instruction, indicating that the vector instruction is complete when the vector address has been translated; and
graduating the vector instruction if the vector instruction is marked complete.

20. (New) The method according to claim 9, wherein the method further includes: marking the vector instruction as complete, wherein marking the vector instruction as

complete includes:

if the vector instruction is not a memory instruction and if the vector instruction does not require scalar operands, indicating that the vector instruction is complete when the vector instruction is dispatched from the scalar processing unit;

if the vector instruction is not a memory instruction but requires scalar operands, indicating that the vector instruction is complete when the scalar operands are available; and

if the vector instruction is a memory instruction, indicating that the vector instruction is complete when the vector address has been translated; and

graduating the vector instruction if the vector instruction is marked complete.